(51) International Patent Classification 6 : G06F 15/177	A1	(11) International Publication Number: WO 97/15886
		(43) International Publication Date: 1 May 1997 (01.05.97)

(21) International Application Number: PCT/US96/16979

(22) International Filing Date: 23 October 1996 (23.10.96)

(30) Priority Data:
08/547,025   23 October 1995 (23.10.95)   US

(60) Parent Application or Grant
(63) Related by Continuation
US   08/547,025 (CON)
Filed on   23 October 1995 (23.10.95)

(71) Applicant (for all designated States except US): CALICO TECHNOLOGY, INC. [US/US]; Suite 1350, 4 North Second Street, San Jose, CA 95113 (US).

(72) Inventor; and
(75) Inventor/Applicant (for US only): PASEMAN, William, G. [US/US]; 15841 Siesta Vista Drive, San Jose, CA 95127 (US).

(74) Agents: WOODWARD, Henry, K. et al.; Townsend and Townsend and Crew L.L.P., 8th floor, Two Embarcadero Center, San Francisco, CA 94111-3834 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, US, UZ, VN, ARIPO patent (KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD AND APPARATUS FOR AUTOMATIC AND INTERACTIVE CONFIGURATION OF CUSTOM PRODUCTS

(57) Abstract

A method for configuring a product (1) from a plurality of selectable components (102-120) includes establishing for each component a list of available classes, defining specific properties for each class of each component, defining constraints among components based on said specific properties, selecting a first plurality of components for a product configuration, identifying each of said first plurality of components as selectable, eliminated, or contradicted based on said constraints, and altering said product configuration to avoid eliminated and contradicted components. The method permits computer aided design of a system which allows interactive participation of the designer in identifying required components for a redesign when an initial design is inoperable.

METHOD AND APPARATUS FOR AUTOMATIC AND INTERACTIVE CONFIGURATION OF CUSTOM PRODUCTS

BACKGROUND OF THE INVENTION

This invention relates generally to the configuration of custom products using selectable components, and more particularly the invention relates to automated configuration which allows interactive participation of a

5   designer.

Suppliers of industrial products often have a large variety of component parts which can be used in various configurations in different applications. Typically, each component has specific properties which can be used

10   advantageously with some other components for a desired application but which make the component incompatible for use with other components. When a new application of the components is being considered, realization of an optimum configuration design can be difficult and time consuming in

15   selecting and assembling the various components. Heretofore, computer programs have been available which can quickly advise the system designer if a proposed design is operable or inoperable. However, if the design is inoperable the necessary redesign and new component specifications have not

20   been readily discernible.

SUMMARY OF THE INVENTION

The present invention is directed to computer automated design of a system which allows interactive

25   participation of the designer in identifying required components for a redesign when an initial design is inoperable. The invention allows product families which might number in the thousands of parts to be easily described in a limited number of expressions. In contrast, thousands of

30   lines of software code would be necessary in describing the

2

individual parts, which requires expensive maintenance each
time product lines are updated.

Briefly, for each component a list of available
classes or types is established. Specific properties for each
5   class are then defined. Constraints in using the various
classes of components with other components are then defined
using the properties as constrained variables, for example, in
the form of Boolean relationships.

An initial product specification is specified using
10   an initial selection of classes of components. Each component
of the initial configuration is then identified as selectable,
eliminated, or contradicted based on the constraints.

A feature of the invention allows the designer to
identify the properties of components which cause any one
15   component to be eliminated or contradicted. This aids the
designer in selecting another class of the component for use
in the product configuration.

The invention and objects and features thereof will
be more readily apparent from the following detailed
20   description and appended claims when taken with the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 illustrates a computer system used to execute
the product design in accordance with the invention.
25   Fig. 2 is a functional block diagram of the computer
system of Fig. 1.

Fig. 3A is a design illustrating concepts of the
invention; and Fig. 3B illustrates one application of the
diagram.
30   Figs. 4A-4D illustrate computer monitor views which
aid in analyzing product configurations.

Fig. 5 is a flow diagram of a product configuration
process in accordance with the invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

The configuration of custom products in accordance with the invention is readily implemented with software in a computer system such as illustrated in Fig. 1. Fig. 1 shows a computer system 1 which includes a monitor 3, screen 5, cabinet 7, keyboard 9, and mouse 11. Mouse 11 may have one or more buttons such as mouse buttons 13. Cabinet 7 houses a floppy disk drive 14, and a hard drive (not shown) that may be utilized to store and retrieve software programs incorporating the present invention. Although a floppy disk 15 is shown as the removable media, other removable tangible media including CD-ROM and tape may be utilized. Cabinet 7 also houses familiar computer components (not shown) such as a processor, memory, and the like.

Fig. 2 shows a system block diagram of a computer system 1 used to execute the software of the present invention. As in Fig. 1, computer system 1 includes monitor 3 and keyboard 9. Computer system 1 further includes subsystems such as a central processor 52, system memory 54, I/O controller 56, display adapter 58, serial port 62, disk 64, network interface 66, and speaker 68. Other computer systems suitable for use with the present invention may include additional or fewer subsystems. For example, another computer system could include more than one processor 52 (i.e., a multi-processor system) or memory cache.

Arrows such as 70 represent the system bus architecture of computer system 1. However, these arrows are illustrative of any interconnection scheme serving to link the subsystems. For example, speaker 68 could be connected to the other subsystems through a port or have an internal direct connection to central processor 52. Computer system 1 shown in Fig. 2 is but an example of a computer system suitable for user with the present invention. Other configurations of subsystems suitable for use with the present invention will be readily apparent to one of ordinary skill in the art.

The software program in accordance with the invention uses constraint-based programming. This is a relatively new branch of artificial intelligence (AI) and is

4

increasingly used for the rapid creation of applications due
to its declarative nature and its relative tractability.
Three key elements of constraint-based programs are
constrained variables, domains, and constraints. Constrained
5   variables are similar to enumerated variables in other
languages. Domains define discrete allowed values that the
constrained variables can assume. For example, there can be a
general class of light called stop light, which can only
assume the values of red, green, and yellow. Assume two such
10   lights, one on Main Street northbound, and one on Main Street
westbound, which become two constrained variables of the type
stop light which are defined over the same domain: red,
green, and yellow.
           Constraints can be Boolean relationships between
15   constrained variables. For example, the relationship (Main
Street northbound < > Main Street westbound) is a constraint
in that the value assumed by the variable Main Street
northbound can never equal the value assumed by the variable
Main Street westbound.
20   The general diagram of the relationships is shown in
Fig. 3A, and the diagram for the stop light class given above
is shown in Fig. 3B.
           The present invention utilizes the implementation of
constraints and how applications can interface with constraint
25   networks. One feature in realizing the invention is the
mapping of constraints to objects. While the prior art
concentrates on defining individual variable values, the
invention integrates constraint based reasoning with an object
system. In particular, constrained variables correspond
30   directly to a class instances. In addition, instead of
individually restricting property values of each instance, the
invention restricts the complete set of values of an instance
at once.
           For example, suppose that a toy class has the
35   properties of name, price, and color. Instead of saying that
the properties are restricted to name, (boat, doll, game),
price ($10, $15, $20) and color (blue, pink, brown); the
invention would define instances of toy as restricted to the

three values <boat, $10, blue>, <doll, $15, pink>, <game, $20,
brown>. This requires the writing of far fewer constraints
than in other prior art systems.

In accordance with another feature of the invention,
a computer monitor view provides a mapping between the domain
of a constrained variable and commonly used UI objects such as
pop-ups, combo boxes, fields, and the like. This allows the
user to quickly create an intuitive user interface with the
program. The essence of the view is simple: one can
visualize allowed values of a class as being a relational
"table" with rows and columns. The user then indicates what
properties are to be mapped to what display choice. Then one
item can be dynamically created in the display for each item
in the table.

Items in the display can have several states:
User Selected - selected explicitly by the user
System Selected - selected explicitly by the system
System Eliminated - eliminated by the system
Item Contradicted - simultaneously selected (user or
    system) and eliminated
Variable Contradicted - two different items are
    simultaneously selected in a variable which allows
    only a single selection
Selectable - neither selected, eliminated, or
    contradicted.
With these concepts of the views in mind, Figs. 4A-4D
illustrate several views for display. Fig. 4A illustrates
allowed values for three classes of toys: boat, game, doll.
Assume that two variables A and B are both in the same state,
boat is selected, game is selectable, and doll is eliminated.
Fig. 4B is a radio button view illustrating this state by
mapping selected items to be "filled in" diamonds, selectable
items to be "non-filled in" diamonds, and eliminated items to
be "grayed out" items. A pop-up view is shown in Fig. 4C, but
whereas Fig. 4B only displayed the name column from the
allowed value table, the user has specified that the pop-up
display both the name and price columns. Here, the pop-up
view maps selected items to be "filled-in" circles, selectable

items to be "non-filled in" circles, and eliminated items to
be "X-ed out" items. Fig. 4D is a view for mapping a
contradiction. When a contradiction occurs the entire control
is usually shaded red.

5   In accordance with a feature of the invention, the
designer is allowed to select grayed out or eliminated items.
When this is done it causes a contradiction and all display
items participating in the contradiction turn red. By showing
all items that participate in the contradiction the designer
10   is guided in all the new choices of components. Additionally,
all variables can have a default value. This allows the
propagation of constraints without search, which in turn
improves the performance of the constrained propagation.
Defaults are picked by the user on a per variable basis, and
15   they can be chosen dynamically by the system based on an
ordering specified by the user (e.g., "boat", "doll", "game").
The system dynamically picks the first item on the list that
has "selectable" for the value of the variable. In defining
constraints, configuration choices heretofore defined as rules
20   are rewritten. For example, a rule may specify "if slots
provided by chassis are greater or equal than slots required
by cards, then signal error". This is converted to a
constraint by simply using the rule condition "slots provided
by chassis must be greater than or equal to slots required by
25   cards."

Consider now an application of the invention using
the concept of containment. Containment is a common product
modeling situation where one has a component or group of
components that fit inside one another. Often the component
30   that houses the other components has certain requirements that
must be accounted. For example, a card cage (a box into which
memory and processing cards are inserted) has a limited number
of card slots. Therefore, a constraint relationship has to be
established to ensure that the number of card slots required
35   by the cards placed in the cage is always equal to or less
than the number of card slots provided by the cage.

First, classes of components must be established. The following table identifies three classes or models of card cages:

Card_cage class

Model number	Slots Provided
Chassis_KL4	4
Chassis_KL8	8
Chassis_IV4	4

Thus the three models of card cage component make up the classes of card cages. Specific brands of card cages that a customer can buy are called "allowed values."

Next, each of the three classes has specific properties. One property of a card cage is how many card slots it provides. In the table above two classes of card cages have four slots each while one class provides eight slots.

One property of memory and processor cards is the number of card slots they require within the card cage. The following are tables for classes of memory cards and processor cards, respectively:

Memory_Card

Model number	Slots Required
Western_4MB	1
Intel_8MB	2
Intel_16MB	4

Processor_Card

Model number	Slots Required
MIPS_040	1
MIPS_050	2
MIPS_060	3

When analyzing properties that involve containment problems, where one class contains another, it is useful to think of properties in terms of what they provide, or what they require.

Instances are the particular models or brands of the product components that make up a class, as noted in the model numbers in the above tables.

Menus can be defined for viewing a class in the computer. These are then used in the system layout.

In defining constraints between components based on
the specific properties, relationships provide the logic
system for the model. Instance to instance relationship is
the most basic kind of relationship. One basic example of an
5   instance to instance relationship is to tell the product model
to automatically select an allowed value of a product
component upon the selection of a master component. For
example, one could set up the system that upon selecting
Chassis_KL4 from the card cage class window, the model
10   automatically selects the MIPS 040 processor card. An
instance to instance relationship like this might be useful to
establish a tie between a component and a cable that is
required by that component, thus ensuring that the cable is
not forgotten in the product package. Another type of
15   instance relationship is elimination. One can set up the
models so that an RHS allowed value is eliminated as an option
upon selection of an LHS allowed value. Although an entire
product model could be constructed based on instance
relationships, it would be very time consuming and require
20   much computer space. This translates to a product model that
is extremely difficult to update. In accordance with the
invention constraint relationships provide a much more
efficient way of establishing ties between class instances.

In building a simple constraint, first consider the
25   nature of the model. Card cages provide a limited number of
card slots. The memory card and processor card must be
provided within the cage. Thus the number of slots provided
by the card cage must be greater than or equal to the total
number of slots required by the memory card and the processor
30   card combined. If the equation is written in standard infix
notation, it looks like the following:

Card_Cage:SlotsPrv>=(Memory_Card:SlotsReq+Processor_Card:SlotsReq)

35   When converting the equation to prefix form, it looks like
this:
(>=Card_Cage:SlotsPrv (+Memory_Card:SlotsReq Processor_Card:SlotsReq))

9

Note that each component (card_cage, memory_card, processor_card) in the equation is described by a property (SlotsPrv or SlotsReq). In building constraint relationships, the properties among product components are related.

5   From the foregoing the basic procedures or product modeling are established. Consider now the concept of compatibility. Compatibility ensures that one component is compatible or usable with another component. Like the containment model, the properties established in a
10   compatibility model can be expressed in terms of Prv (provided) and Req (required). Assume that there are two items each available in different colors. Certain colors are compatible with one item while other colors are not. An item A can be purchased in red, blue, or yellow while item B must
15   be purple, blue, or orange. A model is to be designed that, upon selecting one of the items, will eliminate all colors that are not compatible for that item. Thus, if item A is chosen, the colors red, yellow, and blue are selectable while orange and purple are eliminated. Upon selecting item B,
20   purple, orange, and blue are available while red and yellow are not. Thus certain colors are compatible with certain sets or vice versa.

Item	Color
Item A	blue, red, yellow
25 | Item B | blue, purple, orange |

Note that there are two classes, item and color. Two pop-up menus can be provided in a final model screen as well. From
30   the item class, the user selects either item A or B and the program informs the user which colors are compatible with that selection.

       The first step is to establish the classes for the model. Two classes, color and item are present in this
35   example. Thereafter, the property that needs to be established is color provided. As for the colors, each color provides itself; red provides red, purple provides purple, and orange provides orange. Therefore, the property that needs to be established is color provided. In working with
40   compatibility problems, the majority of time one uses string

property types. As for the class item, each item is
compatible with one of three colors. Therefore, each item
requires three colors as options. The specifying of instances
for the class, color, is simply a matter of listing the total
5   colors available.
        The compatibility relationship is a type of
constraint relationship. A string index (Str-index) tells the
system to take the first part of an equation and search for
that part within the second part of the equation. The first
10   part must be smaller in order that it be contained within the
second part. The final equation for color is:

        (Str-index ?Color:ColorPrv ?Item:ColorReq)

This equation tells the system to search for color within the
list of color options for an item. The equation works
15   forwards and backwards. If a color is chosen first, a
compatible set for that color becomes selectable. If an item
is selected first, a compatible color will be selectable. In
using string index equations, the system starts the search
process from the smallest component of the equation and
20   searches for it within the larger component. Notice that the
property values that are entered under the property color Prv
are all single words (red, yellow, blue, purple and orange).
The system searches for these individual pieces of information
within the larger list of property values for color Req, which
25   for item A is red, yellow, blue (all three together) and for
item B is purple, blue, orange. If the equation were written
with the larger component coming before the smaller one, upon
selecting item A the system would search for the information
"red, blue, yellow" among individual color instances like
30   "red", "orange", or "blue" and would never find a match
because the system reads "red, yellow, blue" as one chunk of
information and can never find it within individual words.
        The invention can be summarized in the flow diagram
of Fig. 5. The algorithm expressed in the flow diagram can be
35   summarized as follows:
        <u>Algorithm</u>
        Define Classes
        Define Properties
        Define Allowed Values
40   Define Variables

Define Constraints
Define Variable/View Associations
Layout Views on Windows
Compile (produce rule file)
5   Run (Have <u>standard rule engine</u> load rule file + 5 state rules)
Interact (Have user interact with standard rule engine through
the User interface)

<u>Define Classes</u>
10   For each class, Enter
- the class's name
- the class's parent class

<u>Define Properties</u>
15   For each property, Enter
- the property's name
- the property's associated class
- the property's type (symbol, string, integer, float, set or
enumeration)
20   - the property's default value

"5 State" - Selectable, Selected, Eliminated, Contradicted,
User Selected

_ 25

        The various cases of inputs and outputs in the
following algorithm are summarized in the following table:

12

```
                        VariableState
                           /
            userInput    avState
               /           /

            userInput    avState
               /           /
input ->allowedValue>-output>dependencies->input->allowedValue>-
   /
otherAvInput                    <-Input from other allowedValues
```

The table below is constructed so that on a per allowedValue, avState, VariableState basis,
rules have no feed-back (e.g., output == selected => input == eliminated on the same group). We
do this to keep from inadvertently "storing state."

			INPUTS			OUTPUTS		
case	MLT	INF	otherAvInput	usrInpt	input	output	avState.src	avState.value
0	y	*	don'tCare	select~select~	select~E~	none	input	selectable
1	y	*	don'tCare	select~select~	select+E~	select	input	select
2	*	*	don'tCare	select~select~	select~E+	eliminate	input	eliminate
3	*	*	don'tCare	select~select~	select+E+	none	input	contradiction
4	y	*	don'tCare	select select~	select*E~	select	userInput	select
5	*	*	don'tCare	select select~	select*E+	select	userInput	selectContradiction
0d	n	n	select~	select~select~	select~E~	none	input	selectable
0a	n	y	select~E+-	select~select~	select~E~	none	input	selectable
0b	n	y	select~E++	select~select~	select~E~	select	input	selectable
0c	n	*?	select+E*	select~select~	select~E~	eliminate	input	select ;sherlock holmes
1a	n	*	select~E*	select~select~	select+E~	select	input	select
1b	n	*	select+E*	select~select~	select+E~	none	input	contradiction
4a	n	*	select~E*	select select~	select*E~	select	userInput	select
4b	n	*	select+E*	select select~	select*E~	select	userInput	selectContradiction

```
select    -1
select*   -0,1
select+   at least 1
select~   0
E*        at least on eliminated input
E+        1 eliminated input

E~        no eliminated input
E+-       at least one of other inputs is not eliminated
E++       is where all other inputs are eliminated
MLT       multiple selectable
Src       source
av        available
```

13

There has been described a computer method and
apparatus for configuring a product from a plurality of
selectable items which facilitates the identification of
eliminated or contradicted items and the altering of product
5   configuration to avoid eliminated and contradicted components.
While the invention has been described with reference to
specific embodiments, the description is illustrative of the
invention and is not to be construed as limiting the
invention. Various modifications and applications may occur
10   to those skilled in the art without departing from the true
spirit and scope of the invention as defined by the appended
claims.

14

WHAT IS CLAIMED IS:

1　　　　　　　1.　　A computer implemented method for configuring a
2　product from a plurality of selectable components comprising
3　the steps of:
4　　　　　　　　a)　　establishing for each component a list of
5　available classes,
6　　　　　　　　b)　　defining specific properties for each class of
7　each component,
8　　　　　　　　c)　　defining constraints among components based on
9　said specific properties,
10　　　　　　　d)　　selecting a first plurality of components for a
11　product configuration,
12　　　　　　　e)　　identifying each of said first plurality of
13　components as selectable, eliminated, or contradicted based on
14　said constraints,
15　　　　　　　f)　　identifying for each eliminated or contradicted
16　component specific properties of other components which
17　contradict or eliminate said each component, and
18　　　　　　　g)　　altering said product configuration to avoid
19　eliminated or contradicted components.

1　　　　　　　2.　　The method as defined by claim 1 wherein
2　step c) includes establishing Boolean relationships between
3　components using said specific properties as constrained
4　variables.

1　　　　　　　3.　　The method as defined by claim 2 wherein said
2　Boolean relationships between components are based on positive
3　constraint definitions of requisite configuration rules.

1　　　　　　　4.　　The method as defined by claim 2 wherein step
2　f) includes identifying selectable components versus
3　eliminated components as specific properties of other
4　components are altered thereby guiding in product
5　reconfiguration.

1   5. The method as defined by claim 4 wherein step
2 f) includes selecting an eliminated component and identifying
3 items which participate in eliminating the component.

1   6. The method as defined by claim 4 wherein in
2 step f) variables have default values, constraints are
3 propagated by selecting a default on a per variable basis.

1   7. The method as defined by claim 6 wherein a
2 computer user selects a default.

1   8. The method as defined by claim 6 wherein a
2 computer system dynamically picks a default based on a class
3 ordering.

1   9. The method as defined by claim 6 wherein
2 step e) includes identifying each component with a radio
3 button map of a filled symbol for selected, non-filled symbols
4 as selectable, and a partially filled symbol as eliminated.

1   10. The method as defined by claim 6 wherein
2 step e) includes identifying each component with a filled
3 symbol for selected, a non-filled symbol as selectable, and an
4 X-symbol as eliminated.

1   11. The method as defined by claim 1 wherein
2 step e) includes identifying each component with a radio
3 button map of a filled symbol for selected, non-filled symbols
4 as selectable, and a partially filled symbol as eliminated.

1   12. The method as defined by claim 1 wherein
2 step e) includes identifying each component with a filled
3 symbol for selected, a non-filled symbol as selectable, and an
4 X-symbol as eliminated.

1   13.   A computer program for use in a computer system
2   for configuring a product from a plurality of selectable
3   components, said program comprising
4   a)   a list of available classes for each component,
5   b)   specific properties for each class of each
6   component,
7   c)   constraints among components based on said
8   specific properties,
9   d)   means operable with said computer system for
10   selecting a first plurality of components based on said
11   specific properties;
12   e)   means operable with said computer system for
13   identifying each of said first plurality of components as
14   selectable, eliminated, or contradicted based on said
15   constraints,
16   f)   means operable with said computer system for
17   identifying for each eliminated or contradicted component
18   specific properties of other components which contradict or
19   eliminate said each eliminated or contradicted component, and
20   g)   means operable with said computer system for
21   altering said product configuration to avoid eliminated or
22   contradicted components.

1   14.   The computer program as defined by claim 13
2   wherein c) constraints include Boolean relationships between
3   components using specific properties as constrained variables.

1   15.   The computer program as defined by claim 14
2   wherein said Boolean relationships between components are
3   based on positive constraint definitions of requisite
4   configuration rules.

1   16.   The computer program as defined by claim 14
2   wherein said f) means identifies selectable components versus
3   eliminated components as specific properties of other
4   components are altered thereby guiding in product
5   reconfiguration.

1   17.   The computer program as defined by claim 16
2   wherein said f) means selects an eliminated component and
3   identifies items which participate in eliminating the
4   component.

1   18.   The computer program as defined by claim 16
2   wherein said f) means variables have default values,
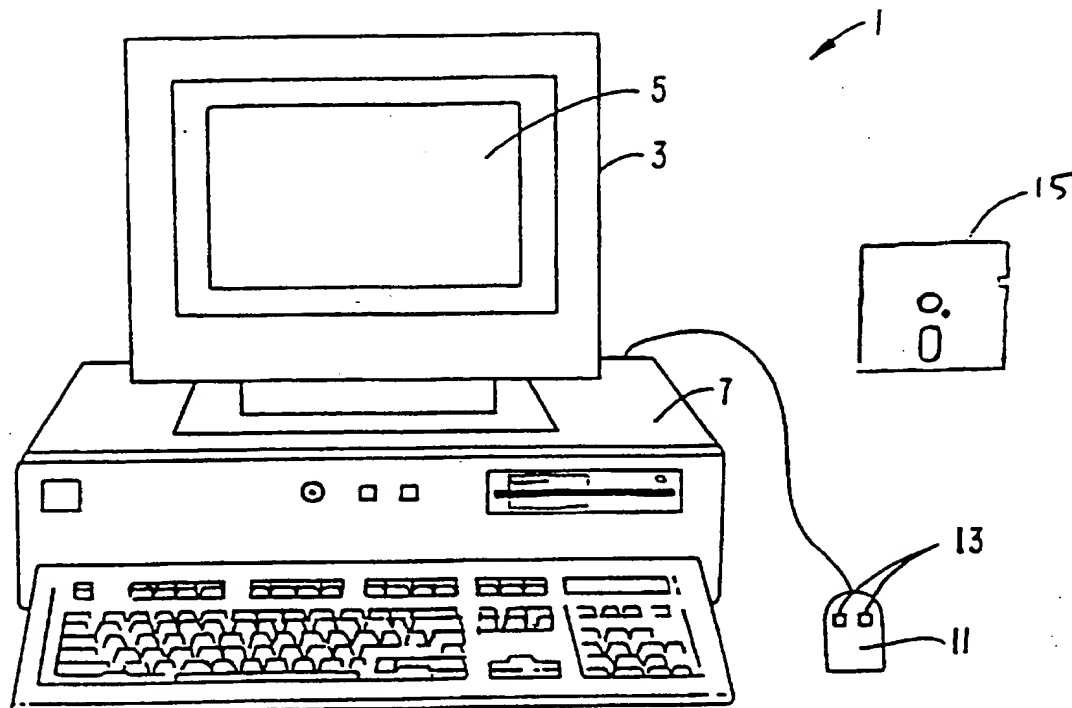3   constraints are propagated by selecting a default on a per
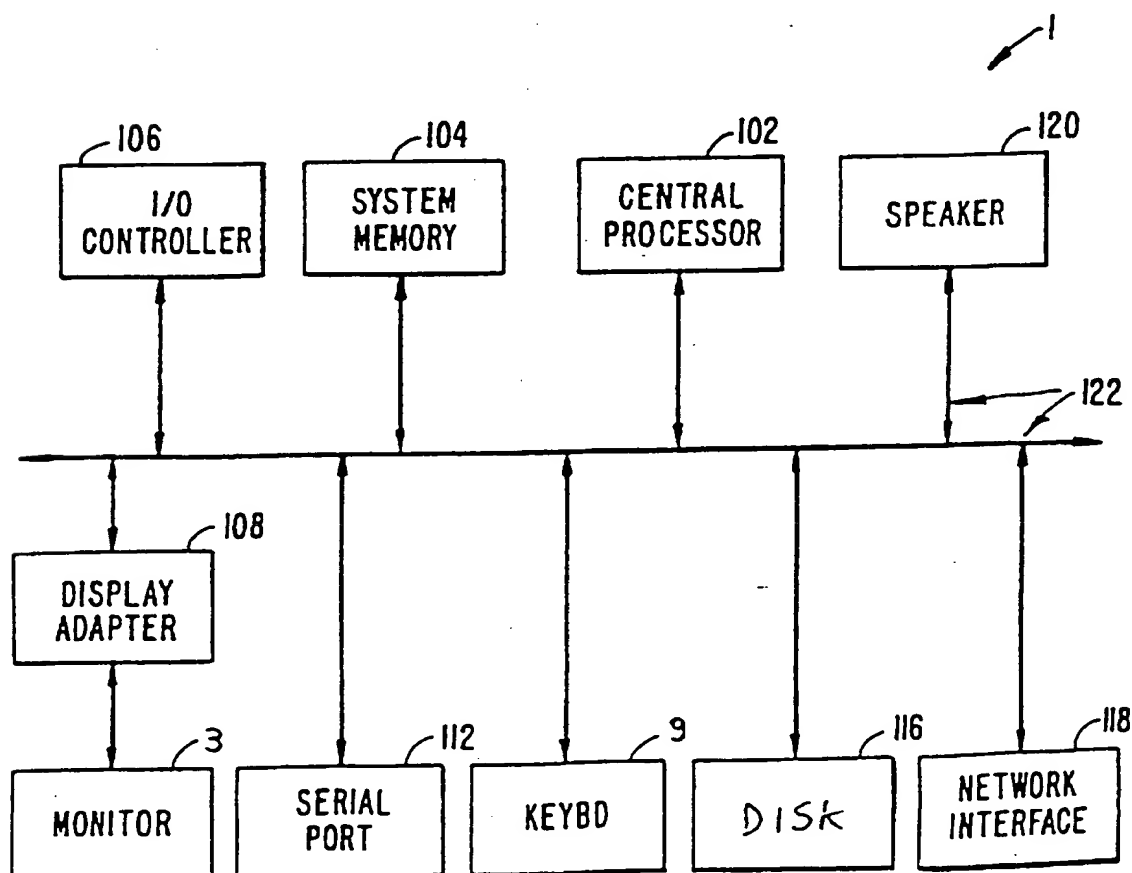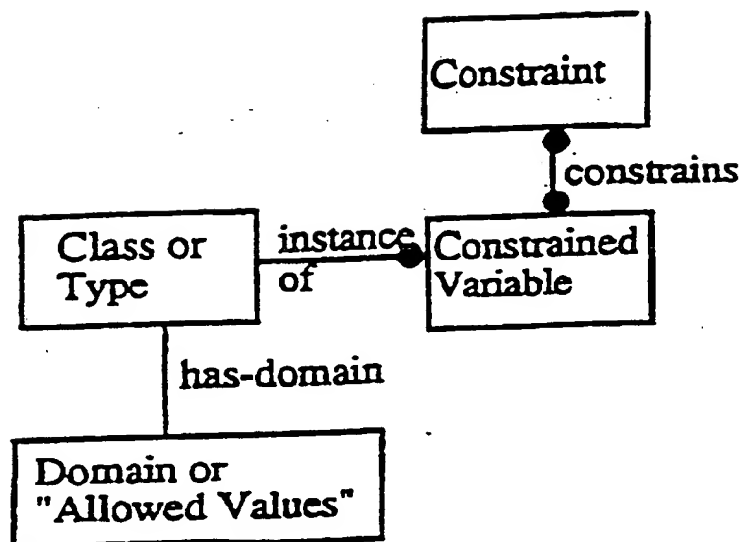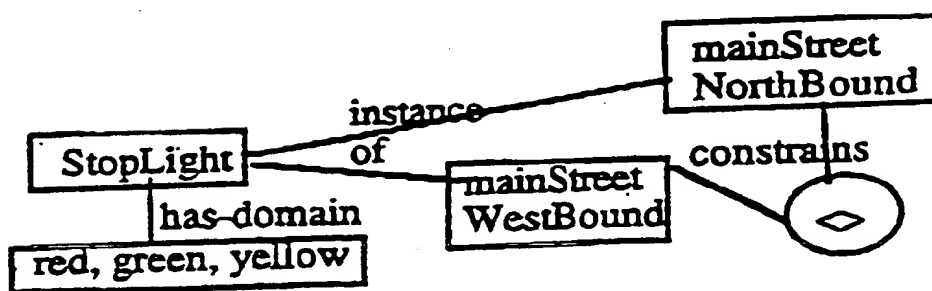4   variable basis.

FIG. 1.

FIG. 2.

FIGURE 3A

FIGURE 3B

allowed values

boat	$10	blue
game	$20	brown
doll	$15	pink

FIGURE 4A

"A"

radio button

◆	boat	selected
◇	game	selectable
◇	doll	eliminated

FIGURE 4B

popup

"B"

boat ▽

selected	●	boat	$10
selectable	O	game	$20
eliminated	✘	doll	$15

FIGURE 4C

popup

none ▽

		boat	$10
!		boat	$10
contradiction	O	game	$20
	!	doll	$15

FIGURE 4D
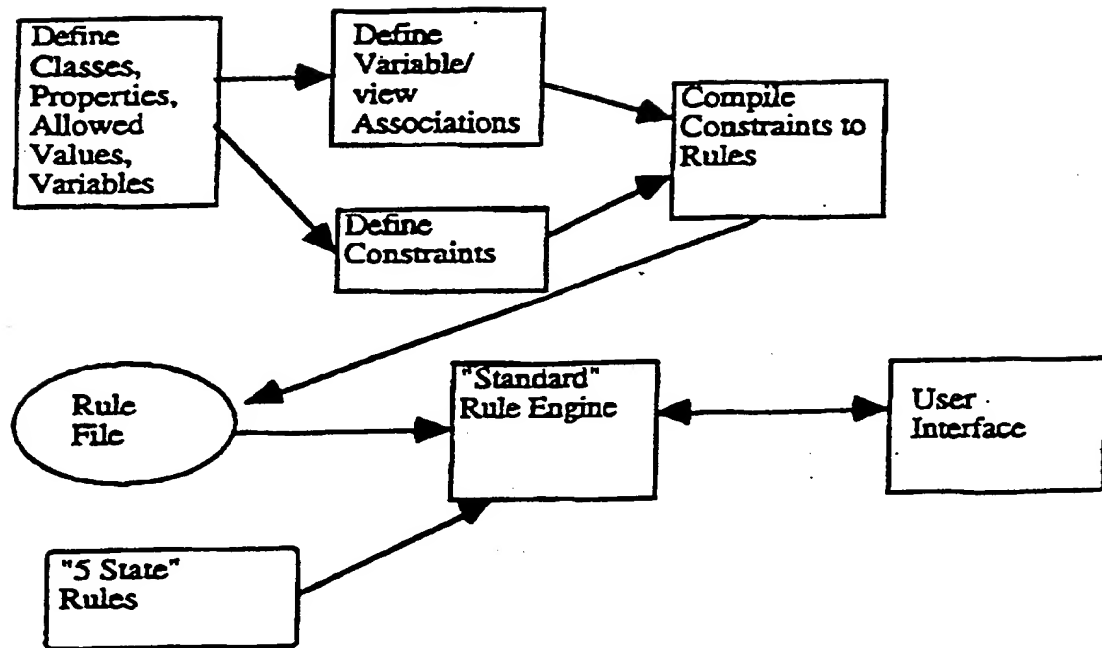
FIGURE 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US96/16979

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :GO6F 15/177

US CL : 395/500,651

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/500,651,700

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS
search terms: configure, component,part,interactive,constraint,conflict

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A,P	US 5,515,524 A (Lynch et al.) 07 May 1996, col. 5- col. 7.	1,13
X	U.S. 5,353,432 A (Richek et al.) 04 October 1994, col.2 lines 9-42.	1,13
Y		2-12,14-18
A	U.S. 5,263,148 (Jones, Jr. et al.) 16 November 1993, col. 2, lines 6-19.	1-18
X	US. 4,591,983 (Bennett et al.) 27 May 1986, col. 4, lines 4-25.	1,13
Y		2-12,14-18

☐ Further documents are listed in the continuation of Box C.   ☐ See patent family annex.

Date of the actual completion of the international search	Date of mailing of the international search report
02 JANUARY 1997	2 7 FEB 1997

Name and mailing address of th ISA/US	Authorized officer
Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Kamal Chaki
Facsimile No. (703) 305-3230	Telephone N . (703) 305-9662

Form PCT/ISA/210 (second sheet)(July 1992)*